LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-202728

# A Short Survey of Document Structure Similarity Algorithms

D. Buttler

March 5, 2004

The 5th International Conference on Internet Computing
Las Vegas, NV, United States
June 21, 2004 through June 24, 2004

# A Short Survey of Document Structure Similarity Algorithms

David Buttler
Lawrence Livermore National Laboratory
Livermore, CA 94550
Email: buttler1@llnl.gov

***Abstract*— This paper provides a brief survey of document structural similarity algorithms, including the optimal Tree Edit Distance algorithm and various approximation algorithms. The approximation algorithms include the simple weighted tag similarity algorithm, Fourier transforms of the structure, and a new application of the shingle technique to structural similarity. We show three surprising results. First, the Fourier transform technique proves to be the least accurate of any of approximation algorithms, while also being slowest. Second, optimal Tree Edit Distance algorithms may not be the best technique for clustering pages from different sites. Third, the simplest approximation to structure may be the most effective and efficient mechanism for many applications.**

## I. Introduction

With the large number of documents on the Web, there is an increasing need to be able to automatically process those documents for information extraction, similarity clustering, and search applications. The majority of work in this area has focused on document content. However, as the Web continues to grow and evolve, more and more information is being placed in structurally rich documents, from HTML to XML. This structural information is an important clue as to the meaning of documents. Identifying documents that are structurally "similar", or structurally "contained" in each other, is an important mechanism to related similar documents that may have sufficiently different content as to make text-based similarity mechanisms nonfunctional.

There are several applications where the structure of the document is critical. Current information extraction algorithms either implicitly or explicitly depend on the structural elements of documents. Structural information can assist in sorting the vast number of pages available from many sites into sets that are roughly comparable. This allows software to separate out the set of documents where information extraction will produce correct results from those documents where the results will not produce useful information.

Structural similarity has been an important topic, and there are algorithms that compute the minimum cost edit distance between any two document structures. However, since these algorithms are expensive, typically requiring $O(n^2)$ or more time to compute the distance, there are opportunities to create algorithms that are faster, but provide slightly less accuracy in computing the distance.

In this paper we present an overview of the current approximation algorithms used to detect structural similarity in Section II. Then we describe a new approximation for structural similarity based on the shingles technique in Section III. Section IV reports on the speed and accuracy of the approximation algorithms with respect to optimal edit distance algorithms. We conclude with a summary of the different algorithm characteristics in Section V.

## II. State of the Art

The study of Structural similarity in tree-based documents has a long history. Early efforts in tree-based change detection come from [1], [2]. More recent results on tree to tree changes focus on creating a minimal edit scripts to convert one tree into another done by Shasha [3], [4], [5] and Chawathe [6], [7]. There have been significant efforts to adapt structure-based similarity computation to specific semi-structured formats, including NiagraCQ [8], Xdiff [9], and Xyleme [10], [11] for XML documents, as well as AIDE [12], [13] and the ChangeDetector™ [14] system for HTML documents.

Previous work on HTML document similarity has generally focused on content similarity, as with page shingles [15]. Current structural similarity has focused on XML schema similarity. DTD similarity [16] focuses on pair-wise similarity between documents with unknown, but similar, DTD's. This requires $O(n^2)$ for each document comparison. Other work has transformed the problem of structural similarity between documents into time-series similarity solved by a Fourier Transform [17], implemented using a Fast Fourier Transform to produce $O(n \times lgn)$ comparisons.

In this paper we introduce a measure of structural similarity applies the shingle technique to the structure of a document. This requires $O(k)$ to create the representation of a single document (where $k$ is the number of nodes), and provides constant time comparison of two documents. The savings in computation comes at a price of a trade-off in accuracy, which can be arbitrarily reduced to tune the technique for different requirements. Section IV shows how this technique compares to other approximation algorithms over different data sets.

### A. Similarity Algorithms

Here we give an overview of the different types of algorithms that have been used to determine document similarity. The different metrics described are tree edit distance similarity, tag similarity, Fourier transforms, and path similarity. The

motivation and algorithms for the shingle technique are given in Section III.

**Tree Edit Distance (TED) Similarity**. Several authors have provided algorithms for computing the optimal edit distance between two trees. This paper follows the dynamic programming implementation described by Nierman and Jagadish [16]. In general, edit distance measures the minimum number of node insertions, deletions, and updates required to convert one tree into another. This can be converted into a similarity metric by normalizing the number of edit operations with the number of nodes in the tree representing the larger document. Let $N_i$ be the set of nodes in the tree representation of document $D_i$. Then,

$$TED(D_i, D_j) = \frac{editDistance(D_i, D_j)}{max(|N_i|, |N_j|)}$$

**Tag Similarity**. Tag similarity is perhaps the simplest metric for structural similarity, as it only measures how closely the set of tags match between two pages. In XML documents, tags are one component of schema; pages that use a similar set of tags will likely have a similar schema. The tag sets of the two documents can be compared to measure their overlap. Let $T_i$ be the set of tags contained in page $D_i$, and $T_j$ be the set of tags contained in page $D_j$. Simple tag similarity of two pages is the intersection of the set of tags from $T_i$ and $T_j$ over the union.

However, this is not satisfactory for several reasons. One critical problem is that pages conforming to the same schema, such as HTML, have only a limited number of different tags; one page may contain a large number of a particular tag, while the comparison page may contain relatively few occurrences of the tag. To compensate for tag frequency, we can introduce a weighted similarity measure. Let $t_{ik}$ be a member of $T_i$, and $w_{ik}$ be the number of times tag $t_{ik}$ appears in $D_i$. Also, let $t_{jk}$ be the corresponding tag in $T_j$, and $v_{jk}$ be the number of times tag $t_{jk}$ appears in $D_j$. If there are $n$ unique tags that occur in pages $D_i$ and $D_j$, then Weighted Tag Similarity (WTS)

$$WTS(D_i, D_j) = \frac{\sum_{k=1}^{n} 2 \cdot min(w_{ik}, v_{jk})}{\sum_{k=1}^{n} (w_{ik} + v_{jk})}$$

As this only involves the set of tags in each document, the accuracy of the structure of the document is independent of the tags used. As a result, this metric should have very low accuracy in environments, like HTML documents, when the tag set is limited, but the structure varies widely. It is likely to be more accurate in repositories of documents that follow a small set of schemas, where the schema limits the variation in structure of documents.

**Fourier Transform Similarity Metric**. Flesca *et al.* introduced the Fourier transform technique as a mechanism to compute similarity between documents [17]. The basic idea is to strip all the information from a document except for its start and end tags, leaving a skeleton that represents the structure. The structure is then converted into a sequence of numbers. The number sequence is then viewed as a time series, and a Fourier transform is applied to convert the data into a set of frequencies. Finally, the distance between two documents is computed by taking the difference of the magnitudes of the two signals.

There are several key components of the algorithm that have a large impact on the results. As originally stated, the encoding of a documents structure uses a unique (sequential) positive integer for each start tag in the document, and the negative of the number for corresponding end tags. Attributes are treated as tags. Note that this implies that to compare two documents, the tag number mapping must be the same for each document. Flesca *et al.* chose a multilevel encoding of a document $d$ as a sequence $[S_0, S_1, ..., S_n]$, where

$$S_i = \gamma(t_i) \times expF(t_i) + \sum_{t_j \in nest_d(t_i)} \gamma(t_j) \times expF(t_j)$$

where $\gamma(t_i)$ is the integer corresponding to the $i^{th}$ tag, $expF(t_i) = B^{maxdepth(D)-l_{t_i}}$ is an exponentiation factor determining the weight of the tag, where $B$ is a fixed base, $maxdepth(D)$ is the maximum depth of the documents being compared, $l_{t_i}$ is the depth of the $i^{th}$ tag, and $nest_d(t_i)$ is the set of ancestors of $t_i$.

The final distance metric between two documents $d_1$ and $d_2$ using the Fourier transform is defined as

$$dist(d_1, d_2) = (\sum_{k=1}^{M/2} (|[FFT(h_1)](k)| - |[FFT(h_2)](k)|)^2)^{\frac{1}{2}}$$

where $FFT$ is the interpolation of the Fast Fourier Transform w.r.t. the frequencies appearing in both $h_1$ and $h_2$, $h_1$ is the signal corresponding to $d_1$, and $M$ is the total number of points appearing in the interpolation.

There are some difficulties with this approach. First, the FFT requires the number of points to be a power of 2. A DFT implementation uses exactly the points in the time series representation of the document, implying that the accuracy of the DFT and FFT approximation algorithms will be different. For our comparisons, we demonstrate only the FFT as the DFT is $O(n^2)$, it takes longer than the TED algorithm in practice, and the accuracy of the DFT approximation algorithm is lower than the FFT.

Second, the requirement to harmonize the tag mapping and precompute the maximum depth of the compared documents precludes the ability to precalculate any part of the algorithm based on a single document to reduce pair-wise comparison time.

Third, Fourier transforms typically operate on a repeating, infinite time series. Documents, as we encounter them, are finite. To apply the transforms, one must extend the time series extracted from document to infinitely repeat. It is not clear what this means in terms of the original document, and the effect this has on the comparison.

## III. Path Shingle

The problem with using an optimal TED algorithm is that it is extremely expensive on large documents. The approximation algorithms presented so far are either unintuitive (Fourier transforms), or provide only a coarse similarity metric (weighted tags). We feel that there needs to be a fast approximation algorithm that can be tuned for accuracy depending on the application.

Shingles were introduced by Broder in [15] as a technique to compare two text documents for similarity and containment. The technique reduced the set of words, or tokens, in a document into a list of hashes that can be directly compared with another document using set difference, union, and intersection to determine similarity or containment.

Further, a subset of the shingles, called a sketch, may be used to determine document similarity. Intuitively, sketches are a random sample of the text in a page. The key is that because the random mapping is constant across all pages, and the results are sorted, the samples are directly comparable across different pages. The overlap in page samples indicates overlap between entire pages.

We show how to adapt this technique, and apply it to the structure of a document. This allows us to essentially compute a linear time approximation to the similarity between the structure of documents. By slightly reducing the accuracy, a constant time comparison between documents of any size may be computed.

### A. Path Similarity

In order to create a structure encoding suitable for applying the shingle technique to, we must find a mechanism to create a list of tokens that represent the structure. Natural choices, such as a depth-first or breadth-first heap encoding [18] prove to be inappropriate due to the unsegmented nature of of the token list used to represent a tree. This means that windows covering a token list cannot distinguish when one branch ends, and the next branch begins. Windows that cover such breaks do not accurately represent any portion of the original structure. Working to build appropriate segmentation of such an encoding leads to another natural encoding: paths.

Semi-structured documents (HTML and XML) can be viewed as a sequence of branches, paths from the root to a leaf. For our purposes, we consider any partial path, the path from the root to any node of the Web document, to be a token. More specifically, it is a list of tag names from the root to the node. A tree can be encoded as a list of these tokens. For example, the simplest tree in HTML has a title and body element, and can be encoded as

```
/html
/html/head
/html/head/title
/html/head/title/[text]
/html/body
/html/body/[text]
```

Path similarity measures the similarity of paths between two different documents. A path is defined as a list of connected nodes starting at the root and terminating in a leaf node. Path similarity can be measured in several different ways: binary, where a path is either equivalent or not; partial, where the number of comparable nodes in each path are discovered; or weighted, where the nodes are weighted according to their distance from the root.

Partial path similarity measures are expensive to compute. Since there are $n!$ possible mappings between the paths of two trees, exhaustive algorithms that produce the optimal similarity score are infeasible. Binary similarity is much cheaper, as each unique path in one version can be matched with its equivalent in the second version of the tree using database join techniques, such as hash joins. Similarity can be computed as the ratio of the matched paths to the total number of paths in the two trees.

### B. Applying Shingles to Paths

A shingle is a contiguous subsequence of tokens taken from a document. Resemblance between documents $D_i$ and $D_j$ is defined as

$$r(D_i, D_j) = \frac{S(D_i, w) \cap S(D_j, w)}{S(D_i, w) \cup S(D_j, w)}$$

where $S(D_i, w)$ is the operator that creates shingles of length $w$ from document $D_i$. Similarly, containment of document $D_i$ in $D_j$ is defined as

$$c(D_i, D_j) = \frac{S(D_i, w) \cap S(D_j, w)}{S(D_i, w)}$$

For convenience and faster processing, shingles may be converted into numbers with a hashing function. This hashing function should provide a high degree of confidence that there will be little or no hash collisions where two shingles map to the same value. Constructing an appropriate hash is made considerably easier by ensuring that the hash space is significantly larger than the shingle space. Depending on the number of tokens in a shingle (or window length), this may be trivial.

A key technique that reduces the complexity of structural comparison to $O(1)$ is to only keep a relatively small sketch of each document. It has been shown [15] that a sampling from a permutation of the set of shingles, chosen uniformly and at random, can be used in an unbiased estimator of resemblance between two documents. One efficient way to achieve this is by applying a pseudo-random number generation algorithm to the hashed values, sorting the results, and choosing only the smallest $k$ of the resulting values.

For applying shingles to the path structure, we define $S(D_i, w)$ as follows: for each node in the tree representation of $D_i$, compute the path from the root to that node; create a hash based on the list of tag names in that (partial) path; add the hash to the current window; slide the window over one (forget the first hash in the window).

Note that by the definition, the set of shingles may be either a set or a bag — analogous to the difference between

tag similarity and weighted tag similarity. It follows that using a set to contain the shingles significantly reduces the expressive power of a shingle and introduces greater error into the approximation.

We have measured the accuracy of the path shingle metric as compared to the partial path metric. The comparisons have been made over differing window sizes using an infinite $k$. The data used for comparisons are typical Web page snapshots downloaded from `my.yahoo.com` over a period of two years. The results show that small window sizes (ranging from one to four) have no effect on accuracy: two clusters have no errors, and four clusters only introduces three percent error. Varying the number of shingles used for comparison did not have any effect on cluster quality. $k$ values tested ranged from ten to 1000, and an unlimited number of shingles. All report the same error to within a tenth of a percent.

## IV. Experiments

In this section we empirically evaluate the accuracy of the different approximation algorithms to the tree edit distance algorithm, and we compare the performance of the different approximation algorithms.

All experiments were run on a dual 2 GHz Xeon processor, 2.4 kernel Linux workstation. The algorithms were written in Java, and executed on the Sun 1.4.2 JVM. All algorithms were implemented using the Page Digest data structure [18], which has been shown to have significant performance benefits over standard DOM tree implementations. Performance measurements were taken as an average over ten runs.

**Comparisons based on clustering**. Clustering is used to asses the quality of different metrics with respect to the tree edit distance (TED) baseline. The TED algorithm produces a provably optimum edit distance between two trees. We assume that this is the best metric of similarity. Other algorithms are expected to produce different different distance measurements that are not directly comparable to an edit distance. However, if a large set of documents are clustered based on a metric, clusters produced by different metrics are directly comparable given that the same clustering algorithm is used. In other words, if two documents are determined to be similar using the TED metric, other metrics should also consider those two documents to be similar, and conversely.

We can then categorize as errors documents that are placed in a single cluster by an approximation metric, but are placed in different clusters by the TED metric. This error measure has some drawbacks. For example, if a set of documents is divided into two clusters, any metric will have a maximum error rate of strictly less than 50%. In general, as the number of clusters, $n$, increases, this maximum error rate is strictly less than $1 - \frac{1}{n}$.

We use two data sets for clustering. First, is a synthetically generated set of 500 XML documents. The set models a book repository, each document listing a single book with its associated author(s), publisher, and publication date. The only structural difference between documents in this set is the number of authors that a book has.

Each metric is used to measure the distance between any two documents. The documents are clustered on these distances using *k-means* clustering. The clusters from each approximation metric is then compared to the clusters from the TED metric, and an error estimation is calculated. The results are shown in Table I. We only test up to six clusters as there are six naturally occurring clusters in this data set, corresponding to the number of authors in a book. The low error rate for the weighted tag metric is to be expected, as the only difference in the structure of these data objects is the number of times the `author` tag appears.

TABLE I
Cluster error rate over book data

| Similarity Metric | Error Rate 6 clusters | 4 clusters | 2 clusters |
|---|---|---|---|
| Weighted Tag | 6% | 5% | 2% |
| FFT | 60% | 46% | 47% |
| Path | 0 % | 0% | 2% |
| Path Shingles | 6% | 5% | 2% |

The second data set is drawn from a set of snapshots taken of the following Web sites: `cnn.com`, `corona.bc.ca`, `news.gnome.org`, `10-10phonerates.com`, and `my.yahoo.com`. The snapshots were taken over a period of two years, between 2001 and 2003, at approximately once per day. Redundant snapshots (as determined by an MD5 signature) were removed, and twenty pages were sampled from each sites snapshot set. The same clustering technique is applied to these documents, only this time we have a predefined cluster (by Web site), and are able to compare each of the algorithms to the predefined cluster. The results are shown in Table II.

TABLE II
Cluster error rate over Web data

| Similarity Metric | Error Rate 6 clusters |
|---|---|
| Weighted Tag | 0% |
| Path | 28% |
| Path Shingles | 34% |
| TED | 38% |
| FFT | 45% |

This error rate seems exceptionally high, especially for the TED algorithm that we have been using for the baseline comparison for the other tests. We speculate that this error rate is caused due to a relatively small vocabulary of HTML structures for displaying content to users. The Path and Path Shingle metrics, while they perform better than TED, also perform worse than would otherwise expected. This may be because they use partial paths to describe the structure of a tree. Deep trees may exhibit a lot of similar structure in the top portions, causing the similarity of two trees from different sites to be

skewed toward similarity with each other. The FFT metrics poor performance does not afford itself to simple explanations. It could be argued that web pages display extremely similar signals, based on the identical first couple of tags (`html`, `head`, and `body`), and similar constructions at the leaf level (lists of tags and text). However, the transformation makes it difficult to determine exactly what features of the Web pages cause them to be seen as so similar.

For a final comparison, we examined snapshots from a single site from the table above. This would be useful in, for example, monitoring a page for changes over time. We chose the `my.yahoo.com` site as the content changes on a regular basis, but the structure only slowly changes over time (for example, when a new graphic is added for particular holidays). Here again we compared the clusters created using the approximation metrics to the TED metric.

TABLE III
CLUSTER ERROR RATE OVER YAHOO! DATA

| Similarity Metric | Error Rate 6 clusters | 4 clusters | 2 clusters |
|---|---|---|---|
| FFT | 33% | 31% | 29% |
| Path | 59% | 46% | 1% |
| Weighted Tag | 60% | 50% | 1% |
| Path Shingles | 61% | 47% | 1% |

Here we observe that while most of the approximation algorithms, with the exception of the FFT algorithm, had a very low error rate at a small cluster size, but the error rate jumped significantly with a larger number of clusters. It is constructive to look at a matrix describing how the mapping varies from the TED metric clustering with the FFT clustering and the Path / Path Single clustering. Table IV describes how the FFT clusters differ from the TED clusters, while Table V describes how the Path clusters differ from the TED clusters.

TABLE IV
CLUSTER COMPARISON BETWEEN FFT AND TED METRICS; 6 CLUSTERS

| Cluster # FFT | TED 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **1** | 1 | 0 | 0 | 0 | 0 | 0 |
| **2** | 1 | 0 | 0 | 0 | 0 | 0 |
| **3** | 1 | 0 | 0 | 0 | 0 | 0 |
| **4** | 1 | 0 | 0 | 0 | 0 | 0 |
| **5** | 92 | 31 | 1 | 8 | 0 | 0 |
| **6** | 1 | 0 | 0 | 0 | 0 | 0 |

From these two matrices, we deduce that the FFT metric tends to cluster all of the data points into a single sample, and thus not providing sufficient discriminating power for very similar Web pages. The Path metric (the Path Shingle and Weighted Tag metrics are essentially the same), on the other hand is more discriminating than the TED metric, and separates pages into more refined groups than the TED. While these experiments record this behavouir as an error, it may have an important function in discriminating trees that are perceivably different, but within equivalent edit distances.

TABLE V
CLUSTER COMPARISON BETWEEN PATH AND TED METRICS; 6 CLUSTERS

| Cluster # Path | TED 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **1** | 26 | 1 | 0 | 0 | 0 | 0 |
| **2** | 18 | 0 | 0 | 0 | 0 | 0 |
| **3** | 12 | 0 | 0 | 0 | 0 | 0 |
| **4** | 18 | 0 | 0 | 0 | 0 | 0 |
| **5** | 0 | 30 | 1 | 8 | 0 | 0 |
| **6** | 23 | 0 | 0 | 0 | 0 | 0 |

The most startling observation is that the weighted tag similarity metric, initially included as a straw man, performs at about the same level of accuracy over the web pages tested as much more sophisticated techniques. This may be attributable to that fact that most of the experiments were conducted over relatively homogeneous pages that are from a single site (as in the book data set or the Yahoo! data set), or that the non-homogeneous sites used a different subset of available HTML tags in order to perform their comparisons. The other observation is that the Fourier transform technique performs poorly over fairly simple data sets. This leads us to conclude that while the idea of converting structure into a 'simpler' format for comparison, it is not an appropriate technique for comparing document structure, either analytically or experimentally.

### A. Performance Comparisons

The main reason to choose an approximation algorithm is to increase speed to an acceptable level in situations where the optimal algorithm is too slow. Document clustering for data extraction, or search and retrieval methods, provides an excellent example for structural similarity. The relatively small dataset used in accuracy estimation demonstrates the effectiveness of approximate similarity computation. Figure 1 shows the relative cost for the different algorithms over a logarithmically increasing number of book documents.

The clustering time was computed on documents of a trivial size, less than a kilobyte. To better understand just the cost of computing the difference between two documents, we compared the cost of each metric over the TPC-H benchmark data. This data was randomly generated by the Toxgene [19], XML document generator. The parameters for the generator were changed so as to produce documents containing 1%, 5%, 10%, 15%, 20%, and 25% of the original data set. The generator was run twice to produce two different documents at each fraction. The results are shown in Figure 2.

Here we see that the TED algorithm is several orders of magnitude slower than any of the approximation algorithms. The FFT algorithm also shows that, even though it has traded a significant amount of accuracy for speed, it is still an order of magnitude slower than the Weighted Tag, or the Path Shingle metrics, both of which are significantly more accurate.

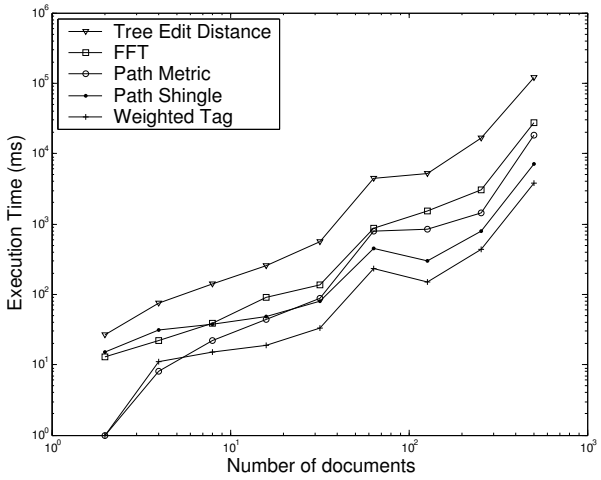


Fig. 1. Cost of clustering book documents using *k-means*

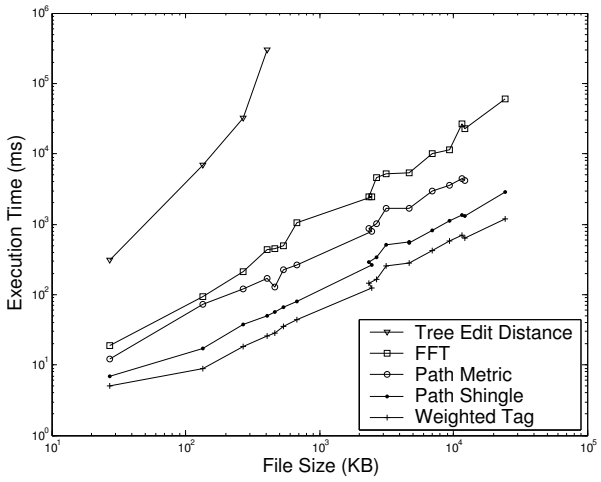


Fig. 2. Pair difference cost of similarity metrics over TPC-H data

**Large File Support**. The optimizations that trade time for memory size work very well for small to moderate sized files, but the Tree Edit Distance algorithm dramatically slows on large files that cause the data structures to exceed physical memory. After physical memory is exhausted, the machine is forced to use swap memory—which is several orders of magnitude slower.

Shingles have the advantage of creating constant-sized fingerprints of large files, eliminating the need to maintain complex data structures in memory when calculating the similarity.

Shingles may also be partially tuned, even after the original fingerprint is taken. Given a set of window hashes, only the top $k$ need to be compared. The number of hashes compared can be adjusted to trade accuracy for speed and memory space. This also allows for more fingerprints to be held in memory at one time for lower accuracy comparisons.

## V. Summary

We have presented several algorithms for measuring document structure similarity, comparing their accuracy and performance. We have several interesting observations.

First, we have provided an experimental critique of the Fourier transform method described in [17]. While a Fourier transformation provides a faster approach to similarity measurements than the optimal tree edit distance algorithm, the approach does not offer an accurate measure of similarity in several types of situations. In addition, the performance of this technique is often poorer than other, more intuitive approaches to approximate similarity.

Second, for many applications of structural similarity, the simplest technique of counting tags provides acceptable accuracy with the best performance. We initially set up the Weighted Tag similarity metric as a straw man to provide the fastest reasonable approximation to structural similarity. However, it turned out to perform as well or better than any of the approximation algorithms. While it does not provide certain structural features that the either the tree edit distance metric (matching of identical subtrees) or the path shingle metric (substructure containment) do, for applications which do not require such features, this algorithm is both fast and reasonably discerning.

Finally, we presented a new similarity metric based on the paths present in a documents structure. We optimized this metric by applying the shingle technique to create constant sized representations of arbitrary documents, allowing clustering techniques to be applied to much larger sets of documents than is possible with other structural similarity measures. In addition this metric presents the ability to search large document sets for substructures, and the capability to do some type of structural mining in sets of tree-based documents.

In the near future we plan to release the source code for each of these algorithms, data, and the testing framework as open source. We hope this will spur the development of superior similarity metrics, and promote these techniques.

## Acknowledgment

The author would like to thank Chuck Baldwin and Ghaleb Abdulla for stimulating conversations on these topics. The author would also like to thank Daniel Rocco for setting up the first experimental framework, and for assistance in developing the Page Digest data structure that was used as the basis for many of these algorithms.

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48. UCRL-CONF-202728.

## References

[1] K. C. Tai, "The tree-to-tree correction problem," *Journal of the ACM*, vol. 26, no. 3, 1979.

[2] S. Y. Lu, "A tree-to-tree distance and its application to cluster analysis," *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1*, no. 2, 1979.

[3] D. Shasha and K. Zhang, "Fast algorithms for the unit cost editing distance between trees," *Journal of Algorithms*, no. 11, 1990.

[4] K. Zhang, D. Shasha, and J. T.-L. Wang, "Approximate tree matching in the presence of variable length don't cares," *J. Algorithms*, vol. 16, no. 1, pp. 33–66, 1994. [Online]. Available: citeseer.nj.nec.com/zhang93approximate.html

[5] D. Shasha and K. Zhang, "Approximate tree pattern matching," in *Pattern Matching Algorithms*. Oxford University Press, 1997, pp. 341–371. [Online]. Available: citeseer.nj.nec.com/95609.html

[6] S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom, "Change detection in hierarchically structured information," in *Proceedings of ACM SIGMOD*, 1996.

[7] S. S. Chawathe and H. Garcia-Molina, "Meaningful change detection in structured data," in *Proceedings of the 1997 ACM SIGMOD*, 1997, pp. 26–37. [Online]. Available: citeseer.nj.nec.com/article/chawathe97meaningful.html

[8] J. Chen, D. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A scalable continuous query system for Internet databases," in *Proceedings of the 2000 ACM SIGMOD*, 2000.

[9] Y. Wang, D. DeWitt, and J.-Y. Cai, "X-Diff: An effective change detection algorithm for XML documents," *International Conference on Data Engineering*, 2003.

[10] A. Marian, S. Abiteboul, G. Cobena, and L. Mignet, "Change-centric management of versions in an XML warehouse," in *The VLDB Journal*, 2001, pp. 581–590. [Online]. Available: citeseer.nj.nec.com/marian00changecentric.html

[11] G. Cobena, S. Abiteboul, and A. Marian, "Detecting changes in XML documents," in *International Conference on Data Engineering*, 2002, pp. 41 –52.

[12] F. Douglis, T. Ball, Y.-F. Chen, and E. Koutsofios, "The AT&T Internet difference engine: Tracking and viewing changes on the Web," in *World Wide Web*, vol. 1, January 1998, pp. 27–44.

[13] Y.-F. Chen, F. Douglis, H. Huan, and K.-P. Vo, "TopBlend: An efficient implementation of HtmlDiff in Java," in *Proceedings of the WebNet2000 Conference*, San Antonio, TX, November 2000.

[14] V. Boyapati, K. Chevrier, A. Finkel, N. Glance, T. Pierce, R. Stokton, and C. Whitmer, "ChangeDetector(TM): A site-level monitoring tool for the WWW," in *WWW2002*, May 2002.

[15] A. Z. Broder, "On the Resemblance and Containment of Documents," in *Proceedings of Compression and Complexity of SEQUENCES 1997*, 1997.

[16] A. Nierman and H. Jagadish, "Evaluating structural similarity in XML documents," *Fifth International Workshop on the Web and Databases*, 2002.

[17] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese, "Detecting structural similarities between XML documents," *Fifth International Workshop on the Web and Databases*, 2002.

[18] D. Rocco, D. Buttler, and L. Liu, "Page Digest for large-scale Web services," in *Proceedings of the IEEE Conference on Electronic Commerce*, June 2003.

[19] D. Barbosa, A. O. Mendelzon, J. Keenleyside, and K. A. Lyons, "Toxgene: An extensible template-based data generator for XML," in *SIGMOD Conference*, 2002. [Online]. Available: citeseer.nj.nec.com/525958.html